Balanced polymorphism and linear lambda calculus

Noam Zeilberger

MSR-Inria Joint Centre

TYPES 2015
Tallinn, 18 May 2015
a pearl theorem
Linear lambda calculus as an extremal case of parametricity:

\[ \lambda x. x(\lambda y. y) : ((\alpha \to \alpha) \to \beta) \to \beta \]
\[ \lambda x. \lambda y. x(y) : (\alpha \to \beta) \to (\alpha \to \beta) \]
\[ \lambda x. \lambda y. y(x(\lambda z. z)) : ((\alpha \to \alpha) \to \beta) \to ((\beta \to \gamma) \to \gamma) \]
\[ \lambda x. \lambda y. x(\lambda z. z(y)) : (((\alpha \to \beta) \to \beta) \to \gamma) \to (\alpha \to \gamma) \]

Every linear lambda term is (simply-)typable, and its \((\beta\eta-)\)normal form is uniquely identified by its principal type.
Mairson asserts this as a “pearl theorem”. I believe that the first proof is due to Mints.

Mints’ key ideas:

1. The principal type of a linear lambda term is **balanced**:

\[
\begin{align*}
\lambda x.x(\lambda y.y) & : \quad ((\alpha \to \bullet \alpha) \to \beta) \to \bullet \beta \\
\lambda x.\lambda y.x(y) & : \quad (\bullet \alpha \to \beta) \to (\alpha \to \bullet \beta) \\
\lambda x.\lambda y.y(x(\lambda z.z)) & : \quad ((\alpha \to \bullet \alpha) \to \beta) \to ((\bullet \beta \to \gamma) \to \bullet \gamma) \\
\lambda x.\lambda y.x(\lambda z.z(y)) & : \quad (((\bullet \alpha \to \beta) \to \bullet \beta) \to \gamma) \to (\alpha \to \bullet \gamma)
\end{align*}
\]

2. Any balanced type (more generally, any balanced typing sequent) has at most one inhabitant up to \(\beta \eta\).

Proof by induction on length of terms.
Mints’ proof is not that complicated, but a pearl theorem deserves a “pearl proof”, and **balanced polymorphism** is a recurring pattern...

Polymorphic **CPS typing**:

\[ \lambda k.k(t) : \forall R. (A \to R) \to \bullet R \]

Semantics of **Separation Logic**:

\[ w \models \phi * \psi \text{ iff } \exists w_1, w_2. (w = \bullet w_1 \otimes \bullet w_2) \land (w_1 \models \phi) \land (w_2 \models \psi) \]

\[ w \models \phi -* \tau \text{ iff } \forall w'. (\bullet w' \models \phi) \supset (w' \otimes w \models \tau) \]

**Ends and coends** in category theory.
I will describe two ways of understanding the pearl theorem:

1. as a simple bijection between string diagrams for linear normal forms and provable balanced sequents, and
2. as a simple bidirectional type inference algorithm.

But first some background...
a graphical language for (neutral/normal) linear lambda terms
Described in a recent paper:

▶ Noam Zeilberger and Alain Giorgetti. A correspondence between rooted planar maps and normal planar lambda terms. To appear in *Logical Methods in Computer Science*.

A rational reconstruction of “lambda-graphs with back-pointers”, and a coloring protocol for *neutral* and *normal* terms.
Dana Scott (1980): pure lambda calculus can be modelled by a reflexive object in a ccc: an object $u$ and morphisms 

$$
\begin{array}{c}
u \\
\xleftarrow{L} \\
u^u \\
\xrightarrow{A}
\end{array}
$$

such that the $L; A = \text{id}_{u^u}$.

Question: what is a model of pure linear lambda calculus?
A monoidal category is a category $C$ equipped with a tensor product and unit operation

\[
\bullet : C \times C \to C \quad I : 1 \to C
\]

associative and unital up to coherent isomorphism.

It is closed if it is also equipped with operations \(\backslash : C^{\text{op}} \times C \to C\) and \(\div : C \times C^{\text{op}} \to C\) right adjoint to the tensor product in each component:

\[
C(y, x \backslash z) \cong C(x \bullet y, z) \cong C(x, z \div y)
\]

It is symmetric if there is a family of isomorphisms

\[
\gamma_{x,y} : x \bullet y \xrightarrow{\sim} y \bullet x
\]

involutive in the sense that \((\gamma_{x,y}; \gamma_{y,x}) = \text{id}_{x \bullet y}\) for all \(x, y \in C\), and which satisfy a few additional equations.
In a smcc, left and right residuals are isomorphic, but let us nonetheless distinguish them and give an explicit name

\[ \sigma_{x,y} : x \setminus y \sim y / x \]

to the isomorphism.

**Definition**

A **linear reflexive object** in a smcc \( C \) is an object \( u \in C \) equipped with a pair of morphisms

\[
\begin{array}{c}
\xymatrix{ u \setminus u & u & u / u \\
\ar[l]^{L} & \ar[r]^{A} & \ar[u]^{L;A} = \sigma_{u,u} }
\end{array}
\]

such that \( L;A = \sigma_{u,u} \).
Idea: recover lambda-graphs by considering a linear reflexive object in a \textit{compact closed category} and applying the machinery of \textit{string diagrams}. 
Recall that any compact closed category has left and right residuals defined by \( x \setminus y \stackrel{\text{def}}{=} \ast x \otimes y \) and \( y / x \stackrel{\text{def}}{=} y \otimes x^\ast \).

The definition of lro translates into the following components in the graphical language of compact closed categories:

\[
\begin{align*}
\ast u \otimes u \xrightarrow{L} u & \sim L \quad & u \xrightarrow{A} u \otimes u^\ast & \sim A \\
L; A = \sigma_{u,u} & \sim =
\end{align*}
\]
Annotating wires with input/output terms:

Some examples:
A coloring protocol for neutral and normal terms

Recall the standard definition of *neutral* and *(β-)*normal terms:

- Any variable $x$ is neutral.
- If $t$ is neutral and $u$ is normal then $t(u)$ is neutral.
- If $t$ is neutral then $t$ is normal.
- If $t$ is normal then $\lambda x.t$ is normal.

Frank Pfenning (TYPES 1993) gave an elegant reformulation of neutral and normal terms as a *refinement type signature*.

Reformulating Pfenning’s reformulation, we introduce the following refinement of the notion of linear reflexive object:

**Definition**

A **linear reflexive pair** in a smcc $\mathcal{D}$ is a pair of objects $B, R \in \mathcal{D}$ equipped with a quadruple of morphisms

\[
\begin{array}{ccc}
  B \backslash R & \xrightarrow{\ell} & R & \xleftarrow{c} & B & \xrightarrow{a} & B / R \\
  \downarrow s & & \downarrow & & \downarrow & & \\
\end{array}
\]

such that $s; c = \text{id}_B$ and $\ell; c; a = (\text{id}_B \backslash c); \sigma_{b,b}; (\text{id}_B / c)$. 

\[ \lambda x.t \]

\[ (\lambda x.t)(u) = t[u/x] \]

\[ t(u) = t \]
Any neutral or normal linear term (with \( i \) free variables) can be given a colored string diagram of the form

\[
\begin{array}{c}
  x_i \cdots x_1 \\
  \cdots \\
  \pi \\
  t \\
\end{array}
\quad \text{or} \quad
\begin{array}{c}
  x_i \cdots x_1 \\
  \cdots \\
  \pi \\
  t \\
\end{array}
\]

which moreover is \underline{free of} \textbf{c-nodes} (= no red boxes).
relating normal linear terms and balanced principal types
Reverse the orientation of blue wires

and replace each blue box (s-node) by a distinct type variable...
\[
\lambda y. yx \leftrightarrow (\alpha \rightarrow \beta) \rightarrow \beta
\]
\[
\lambda y. y(x(\lambda z. z)) \leftrightarrow (\alpha \to \alpha) \to \beta \to (\beta \to \gamma) \to \gamma
\]
\[\lambda y. x(\lambda z.zy)\]
bidirectional type inference
Two moded typing judgments:

\[ \Gamma \leftarrow R \leftarrow A \quad \text{checking against } A, \ R \text{ synthesizes context } \Gamma \]
\[ \Gamma \leftarrow N \Rightarrow A \quad N \text{ synthesizes type } A \text{ and context } \Gamma \]

Well-modeled inference rules:

\[ x : A \leftarrow x \leftarrow A \]
\[ \Gamma \leftarrow R \leftarrow A \Rightarrow B \quad \Delta \leftarrow N \Rightarrow A \]
\[ \Gamma, \Delta \leftarrow R(N) \leftarrow B \]

\[ \Gamma \leftarrow R \leftarrow \alpha \quad \alpha \text{ fresh} \]
\[ \Gamma \leftarrow R \Rightarrow \alpha \]
\[ x : A, \Gamma \leftarrow N \Rightarrow B \]
\[ \Gamma \leftarrow \lambda x.N \Rightarrow A \Rightarrow B \]
This is just dual to standard bidirectional type checking!

\[ \Gamma \leftarrow R \leftarrow A \leftrightarrow \Gamma \Rightarrow R \Rightarrow A \]
\[ \Gamma \leftarrow N \Rightarrow A \leftrightarrow \Gamma \Rightarrow N \leftarrow A \]
todo list
- Formal meaning of “reverse the blue arrows”
- Understanding normalization and type annotations
- Pure lambda calculus and intersection types